#include <stdio.h>

// Structure to represent a process

struct Process {

int pid; // Process ID

int burstTime; // Burst time of the process

int arrivalTime; // Arrival time of the process

int remainingTime; // Remaining burst time of the process

};

// Function to sort processes based on burst time (SJF)

void sortProcesses(struct Process processes[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (processes[j].burstTime > processes[j + 1].burstTime) {

// Swap processes

struct Process temp = processes[j];

processes[j] = processes[j + 1];

processes[j + 1] = temp;

}

}

}

}

// Function to simulate Shortest Job First (SJF) scheduling

void sjf(struct Process processes[], int n) {

int currentTime = 0; // Current time

int completedProcesses = 0; // Number of completed processes

// Sort processes based on arrival time

sortProcesses(processes, n);

while (completedProcesses < n) {

int shortestJobIndex = -1;

int shortestJobTime = 999999; // Assume a large value for initial comparison

for (int i = 0; i < n; i++) {

if (processes[i].arrivalTime <= currentTime && processes[i].remainingTime > 0) {

if (processes[i].burstTime < shortestJobTime) {

shortestJobIndex = i;

shortestJobTime = processes[i].burstTime;

}

}

}

if (shortestJobIndex == -1) {

currentTime++;

} else {

// Execute the shortest job

processes[shortestJobIndex].remainingTime--;

currentTime++;

// Check if the process is completed

if (processes[shortestJobIndex].remainingTime == 0) {

completedProcesses++;

int turnaroundTime = currentTime - processes[shortestJobIndex].arrivalTime;

printf("Process %d completed. Turnaround time: %d\n", processes[shortestJobIndex].pid, turnaroundTime);

}

}

}

}

int main() {

// Number of processes

int n;

printf("Enter the number of processes: ");

scanf("%d", &n);

// Array of processes

struct Process processes[n];

// Input arrival time and burst time for each process

for (int i = 0; i < n; i++) {

processes[i].pid = i + 1;

printf("Enter arrival time for process %d: ", processes[i].pid);

scanf("%d", &processes[i].arrivalTime);

printf("Enter burst time for process %d: ", processes[i].pid);

scanf("%d", &processes[i].burstTime);

processes[i].remainingTime = processes[i].burstTime;

}

// Run SJF scheduling

sjf(processes, n);

return 0;

}

OUTPUT:-

Enter the number of processes: 4

Enter arrival time for process 1: 5

Enter burst time for process 1: 6

Enter arrival time for process 2: 4

Enter burst time for process 2: 2

Enter arrival time for process 3: 6

Enter burst time for process 3: 5

Enter arrival time for process 4: 6

Enter burst time for process 4: 5

Process 2 completed. Turnaround time: 2

Process 3 completed. Turnaround time: 5

Process 4 completed. Turnaround time: 10

Process 1 completed. Turnaround time: 17

=== Code Execution Successful ===